CAP6135: Malware and Software Vulnerability Analysis

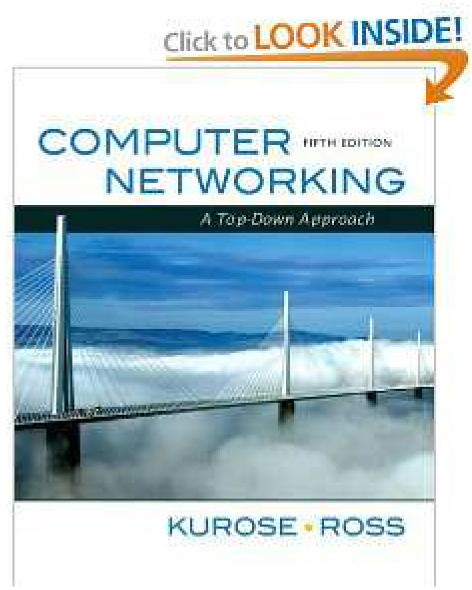
Basic Knowledge on Computer Network Security

Cliff Zou

Spring 2014

Acknowledgement

- This lecture note is modified from the slides provided by textbook:
  - *Computer Networking: A Top Down Approach Featuring the Internet*, J. Kurose & K. Ross, Addison Wesley, 5rd ed., 2009

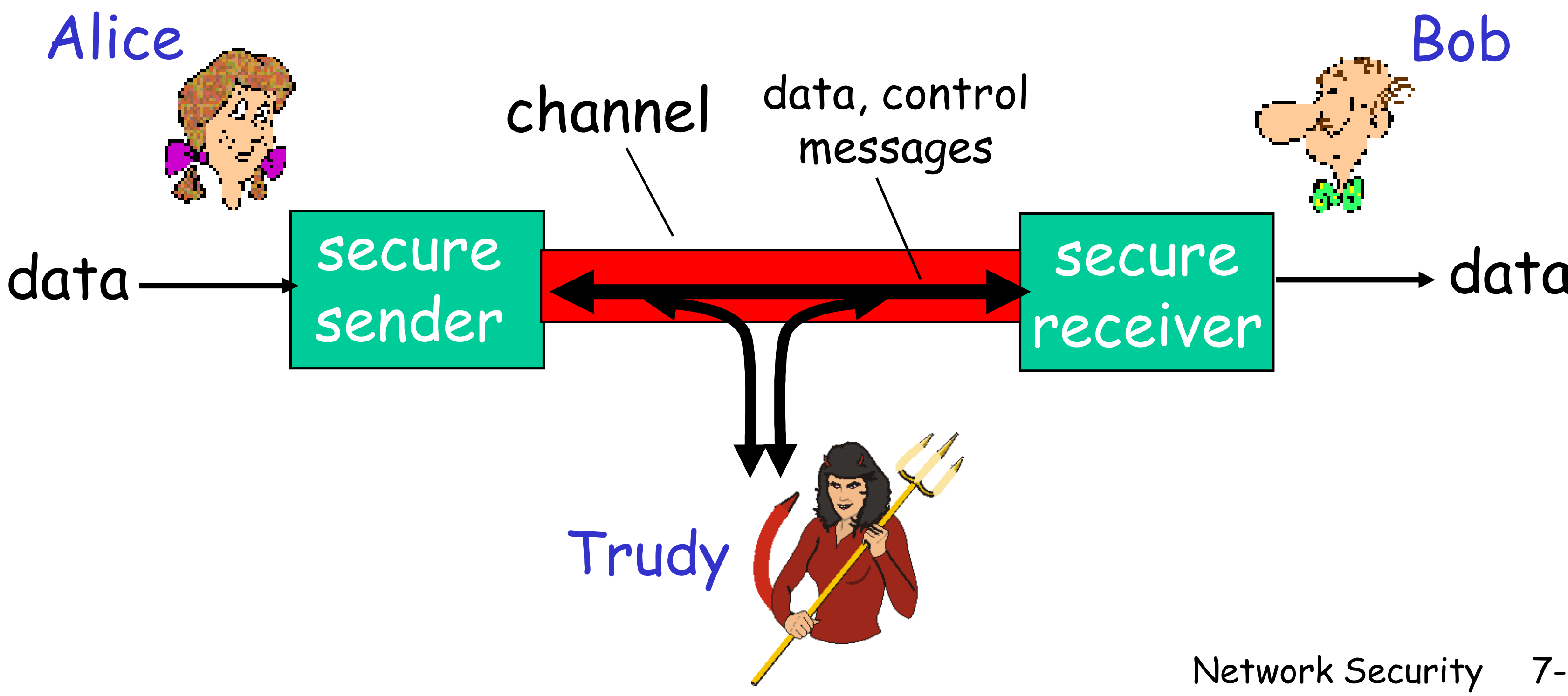


Why This Introduction?

- Some students may have no knowledge of basic cryptography and basic network security
  - Such knowledge is important
  - Such knowledge is necessary for continuing learning malware and software security

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

- Web client/server (e.g., on-line purchases)
- DNS servers
- routers exchanging routing table updates
- Two computers in peer-to-peer networks
- Wireless laptop and wireless access point
- Cell phone and cell tower
- Cell phone and bluetooth earphone
- RFID tag and reader
- ....

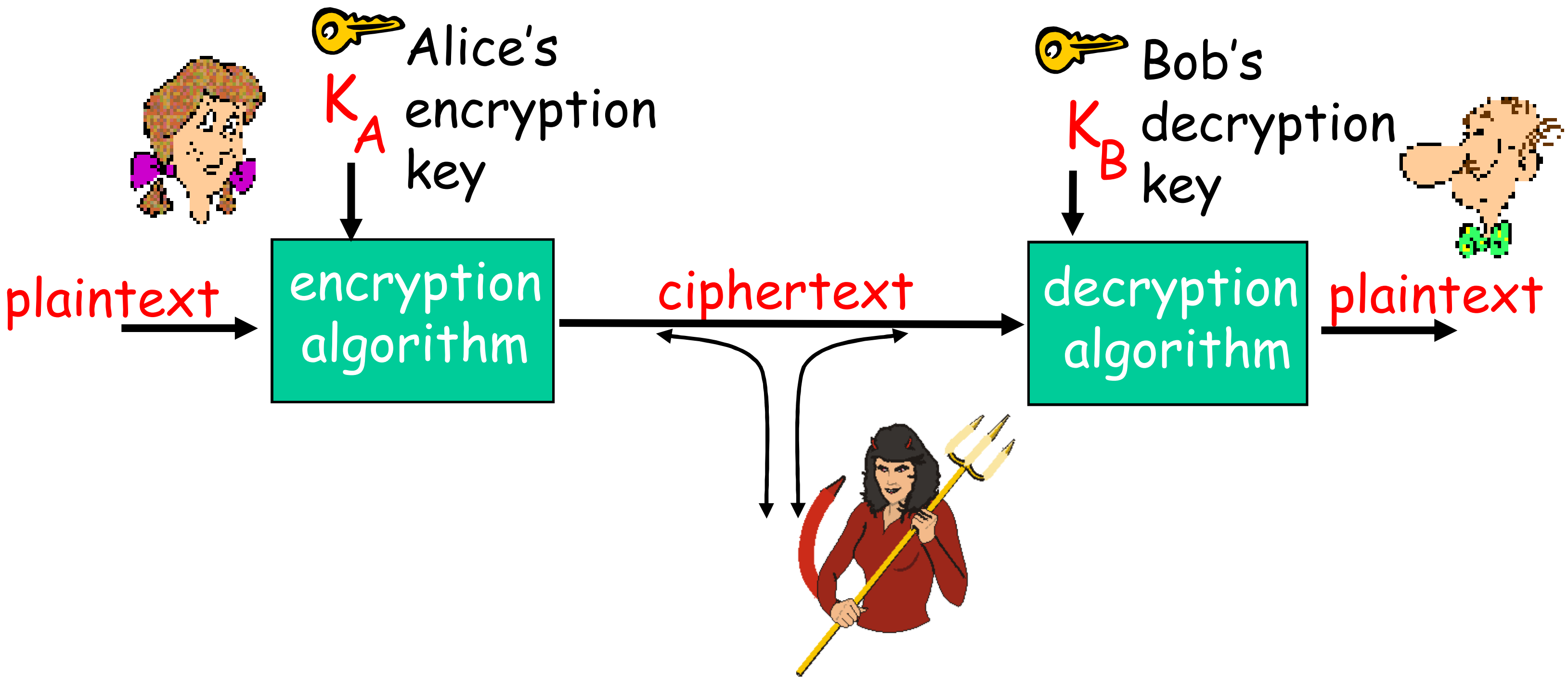
There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: a lot!

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

The language of cryptography



symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

Classical Cryptography

- Transposition Cipher

- Substitution Cipher
  - Simple substitution cipher (Caesar cipher)
  - Vigenere cipher
  - One-time pad

Transposition Cipher: rail fence

- Write plaintext in two rows in column order
- Generate ciphertext in row order

- Example: "HELLOWORLD"

HLOOL
ELWRD

ciphertext: HLOOLELWRD

Problem: does not affect the frequency of individual symbols

Simple substitution cipher

substituting one thing for another

- Simplest one: monoalphabetic cipher:
  - substitute one letter for another (**Caesar Cipher**)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

⇓

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

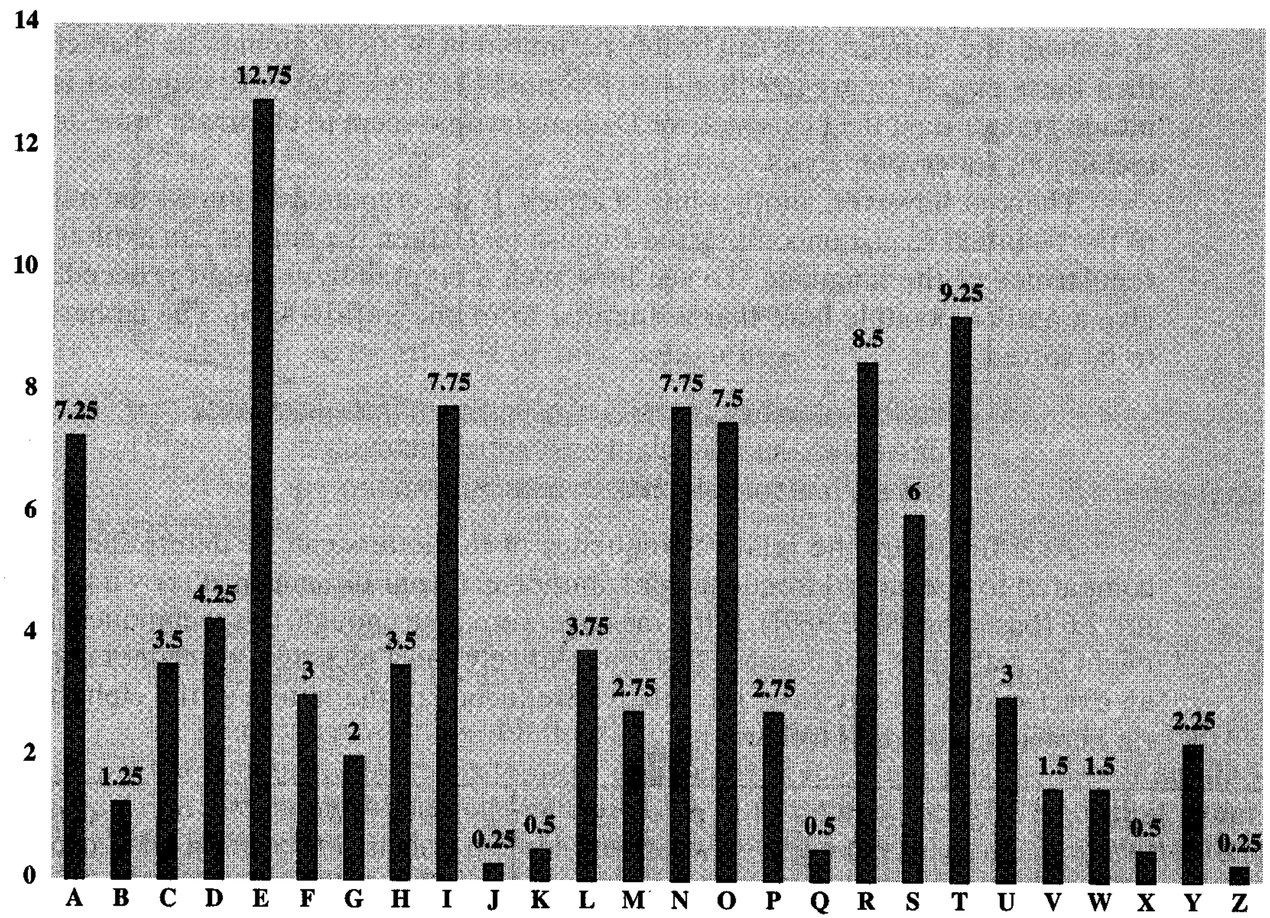
Example: encrypt "I attack"

Problem of simple substitution cipher

- The key space for the English Alphabet is very large: $26! \approx 4 \times 10^{26}$

- However:
  - Previous example has a key with only 26 possible values
  - English texts have statistical structure:
    - the letter “e” is the most used letter. Hence, if one performs a frequency count on the ciphers, then the most frequent letter can be assumed to be “e”

Distribution of Letters in English



Frequency analysis

Vigenere Cipher

- Idea: Uses Caesar's cipher with various different shifts, in order to hide the distribution of the letters.
- A key defines the shift used in each letter in the text
- A key word is repeated as many times as required to become the same length

Plain text:	I a t t a c k	
Key:	2 3 4 2 3 4 2	(key is "234")
Cipher text:	K d x v d g m	

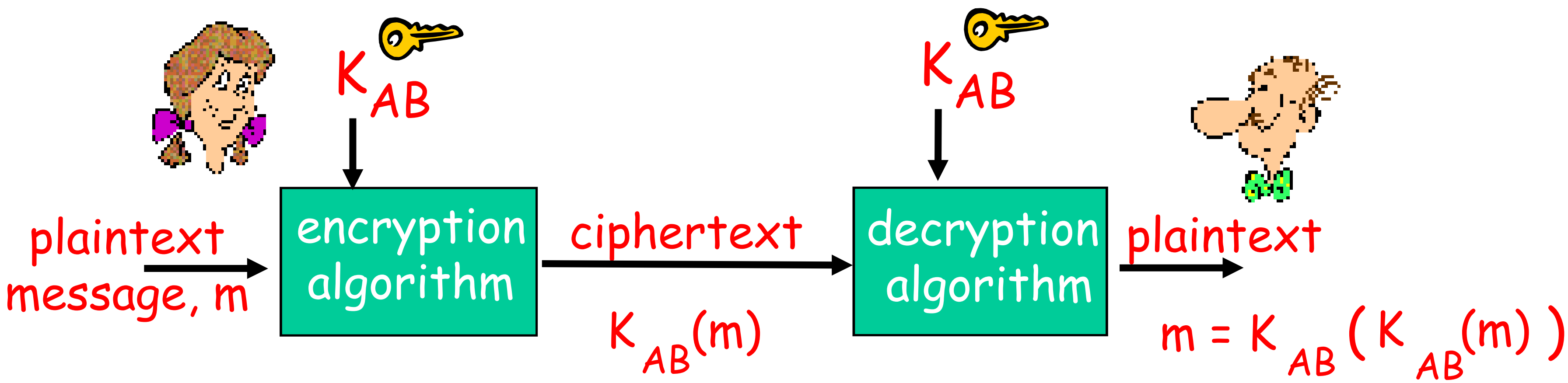
Problem of Vigenere Cipher

- Vigenere is easy to break (Kasiski, 1863):
- Assume we know the length of the key. We can organize the ciphertext in rows with the same length of the key. Then, every column can be seen as encrypted using Caesar's cipher.
- The length of the key can be found using several methods:
  - 1. If short, try 1, 2, 3, ...
  - 2. Find repeated strings in the ciphertext. Their distance is expected to be a multiple of the length. Compute the gcd of (most) distances.
  - 3. Use the index of coincidence.

One-time Pad

- Extended from Vigenere cipher
- Key is as long as the plaintext
- Key string is random chosen
  - Pro: Proven to be "perfect secure"
  - Cons:
    - How to generate Key?
    - How to let bob/alice share the same key pad?
  - Code book

Symmetric key cryptography



symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{AB}

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: how do Bob and Alice agree on key value?

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase ("Strong cryptography makes the world a safer place") decrypted (brute force) in 4 months
  - no known "backdoor" decryption approach
- making DES more secure (3DES):
  - use three keys sequentially on each datum
  - use cipher-block chaining

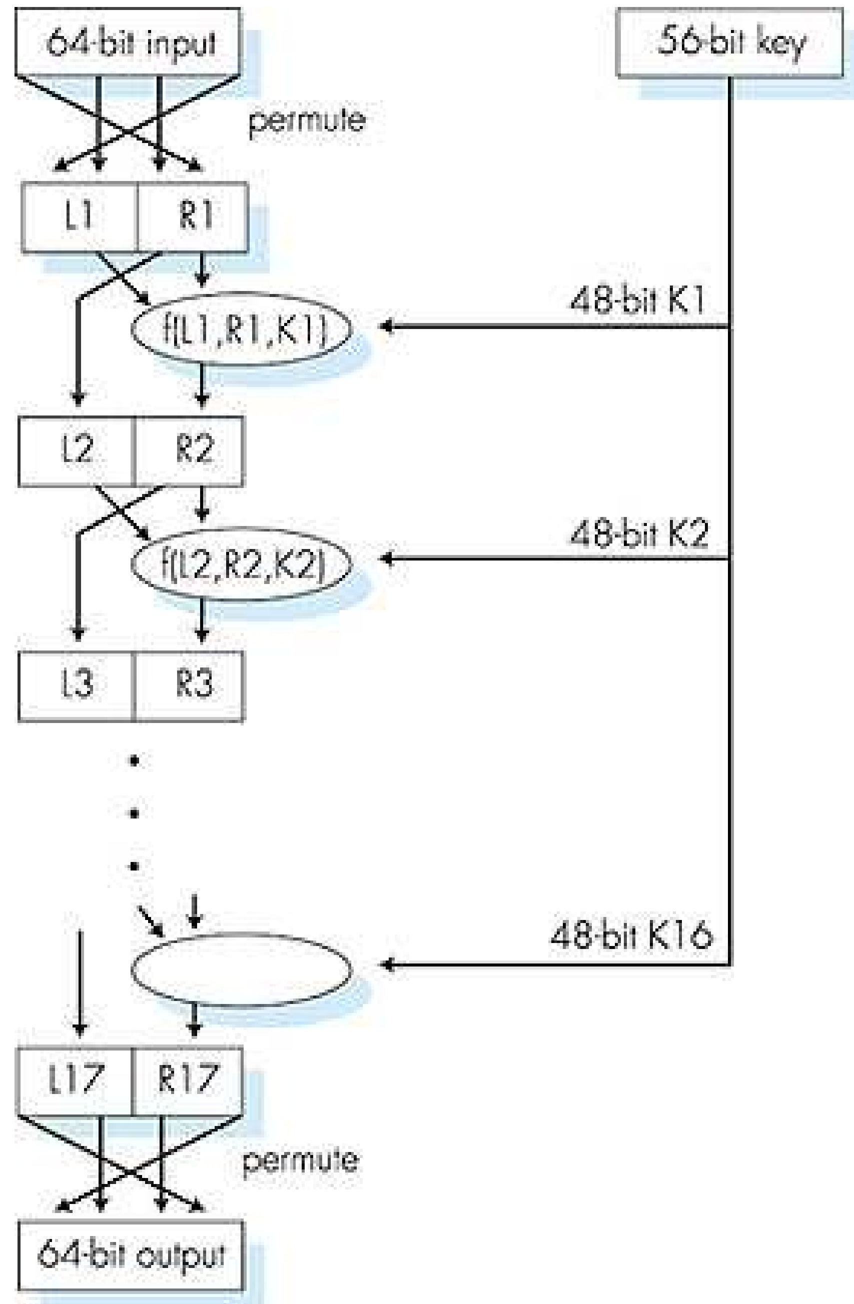
Symmetric key crypto: DES

DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

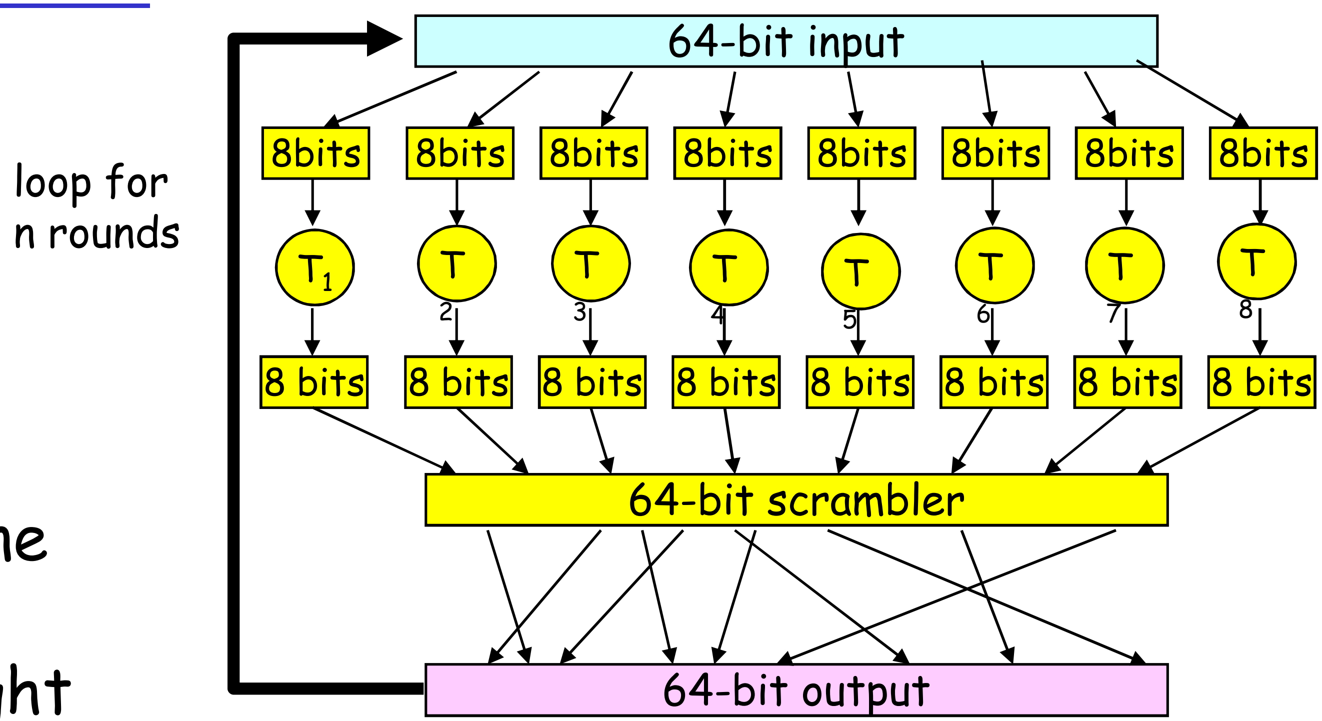
final permutation



AES: Advanced Encryption Standard

- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

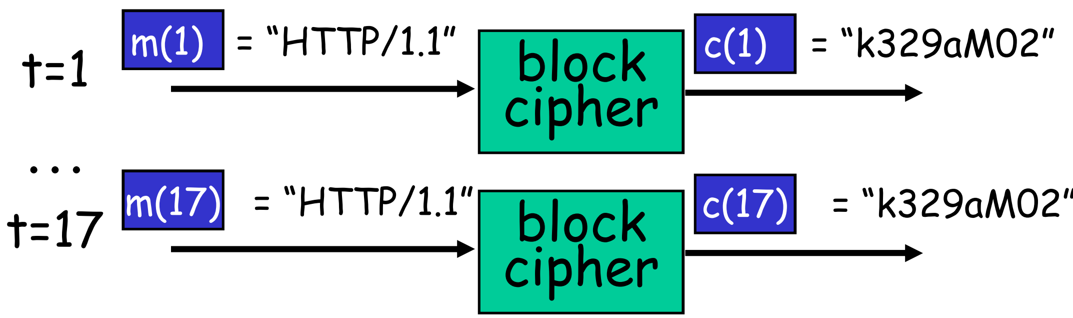
Block Cipher



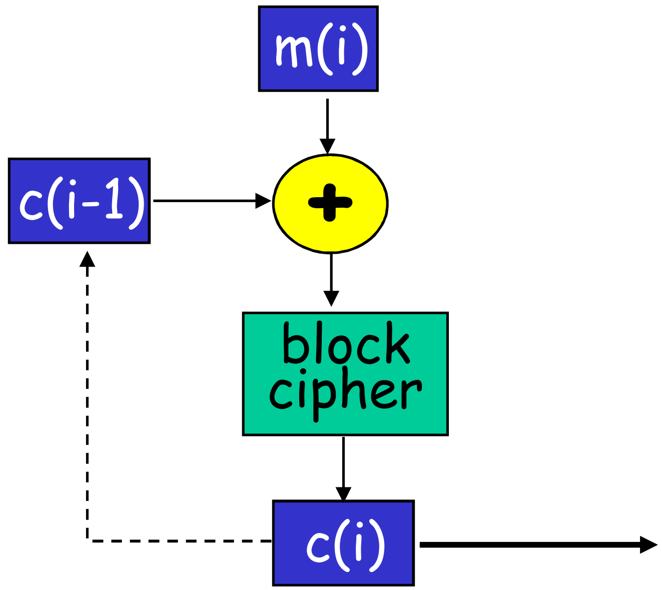
- one pass through: one input bit affects eight output bits
- multiple passes: each input bit affects most output bits
- block ciphers: DES, 3DES, AES

Cipher Block Chaining

- cipher block: if input block repeated, will produce same cipher text:



- *cipher block chaining:* XOR ith input block, m(i), with previous block of cipher text, c(i-1)
  - c(0) transmitted to receiver in clear
  - what happens in "HTTP/1.1" scenario from above?

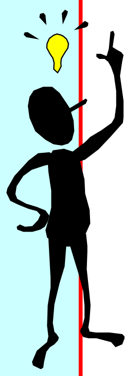


Public Key Cryptography

symmetric key crypto

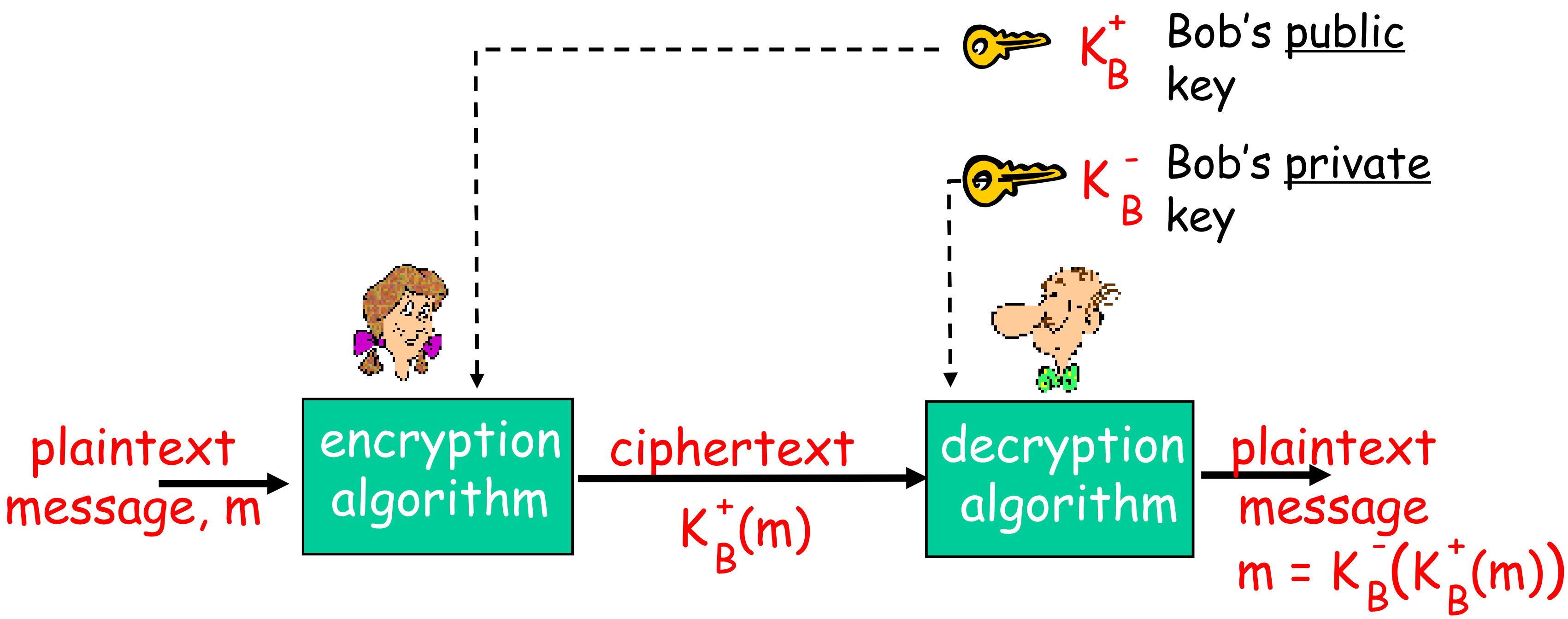
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key cryptography



- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public key cryptography



Public key encryption algorithms

Requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

RSA: Choosing keys

1. Choose two large prime numbers $p, q.$
   (e.g., 1024 bits each)

2. Compute $n = pq$, $z = (p-1)(q-1)$

3. Choose e (with $e<n$) that has no common factors with z. (e, z are "relatively prime").

4. Choose d such that $ed-1$ is exactly divisible by z.
   (in other words: $ed \bmod z = 1$).

5. *Public* key is $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute
   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute
   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

RSA example:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.

$e=5$ (so e, z relatively prime).

$d=29$ (so $ed-1$ exactly divisible by z).

encrypt:	letter	m	m^e	$c = m^e \bmod n$
	l	12	1524832	17

decrypt:	c	c^d	$m = c^d \bmod n$	letter
	17	481968572106750915091411825223071697	12	l

Computational extensive

RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

Result is the same!

Digital Signatures

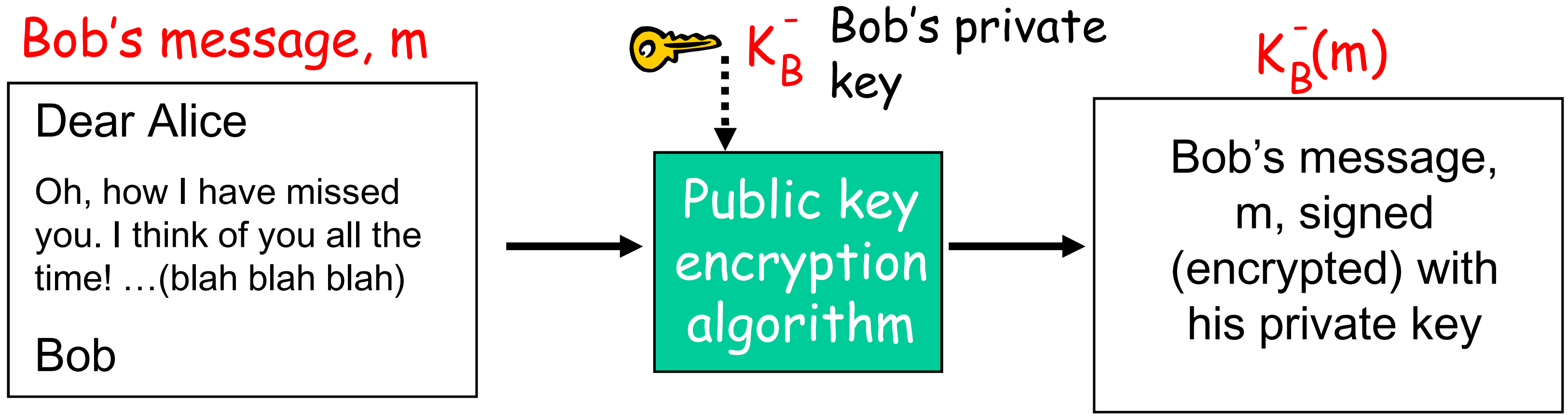
Cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital Signatures (more)

- Suppose Alice receives msg m, digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

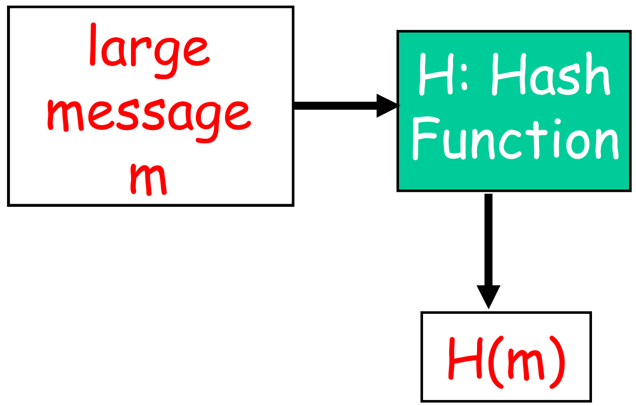
Alice thus verifies that:

- Bob signed m.
- No one else signed m.
- Bob signed m and not m'.

Non-repudiation:

- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m.

Message Digests



Computationally expensive to public-key-encrypt long messages

Goal: fixed-length, easy-to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m)*.

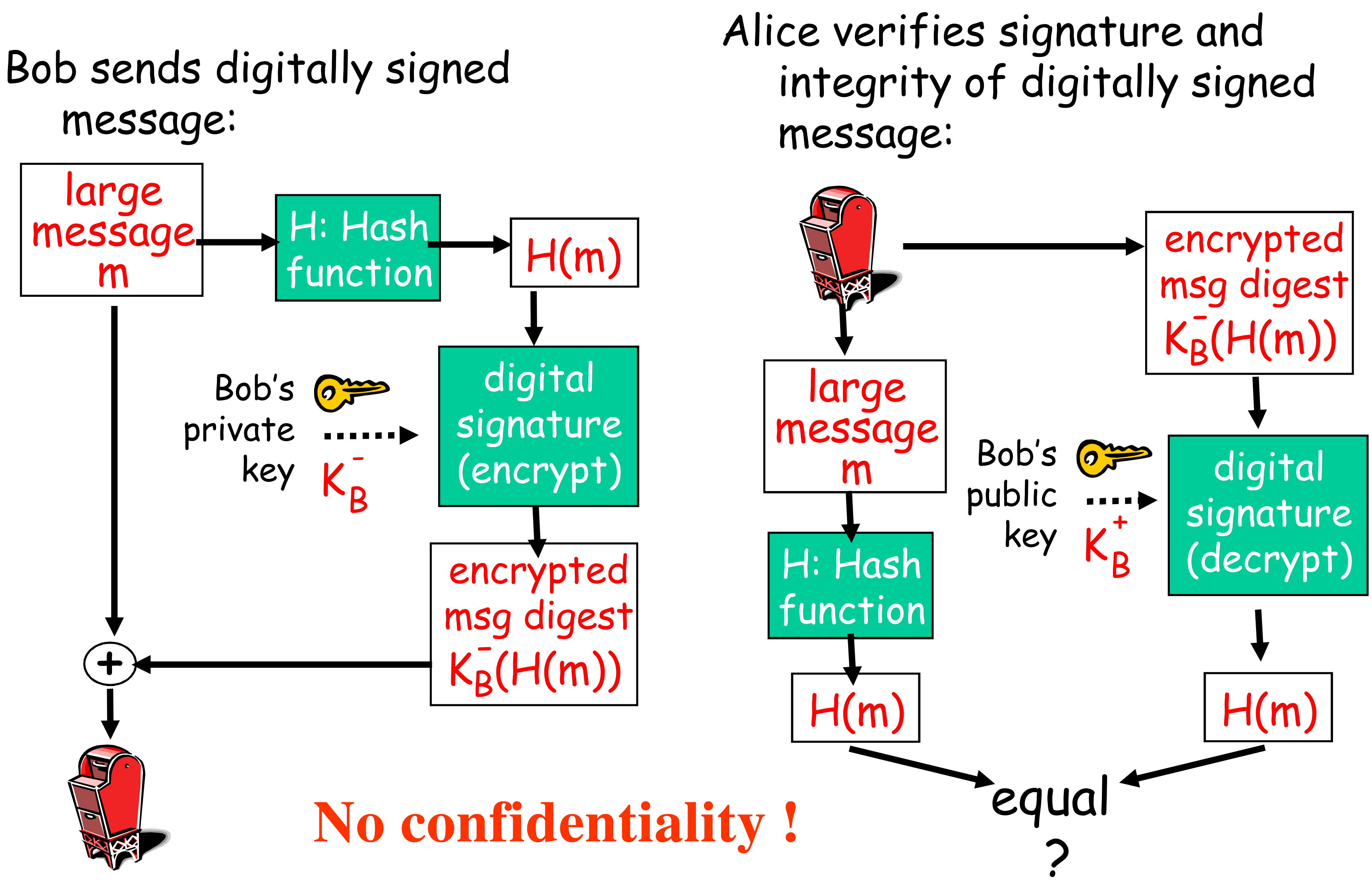
Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Digital signature = signed message digest



No confidentiality !

Trusted Intermediaries

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

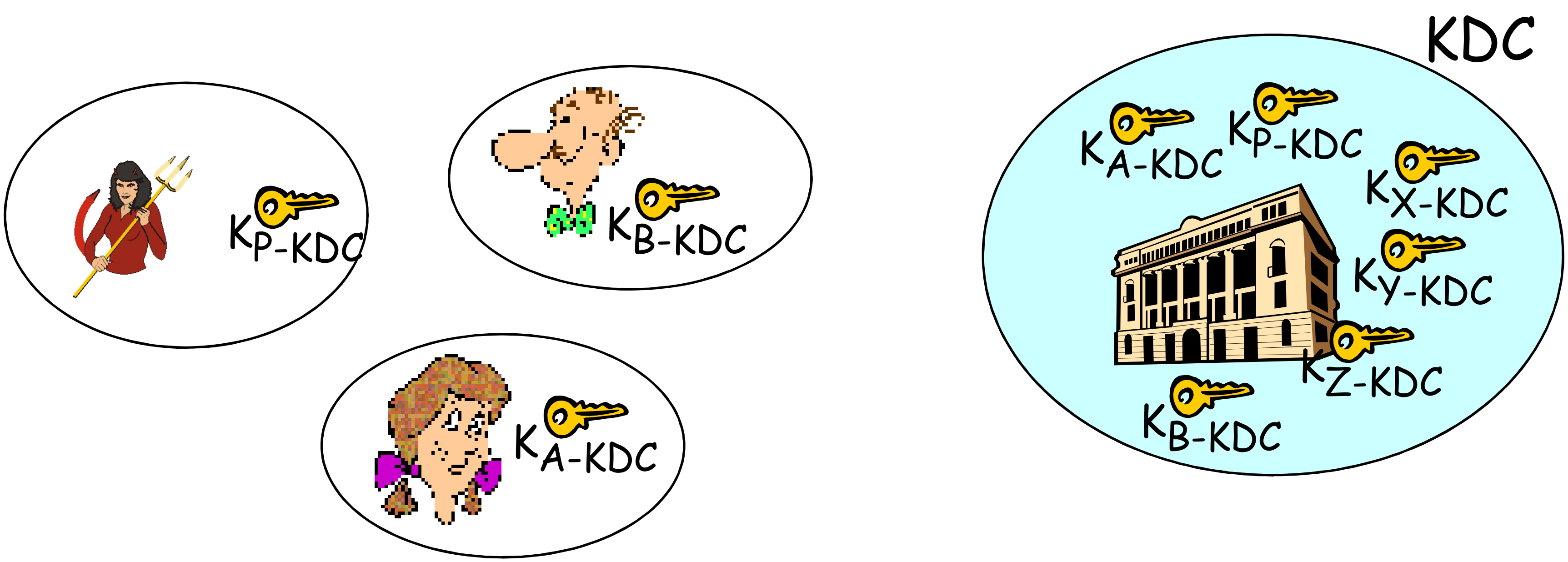
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- trusted certification authority (CA)

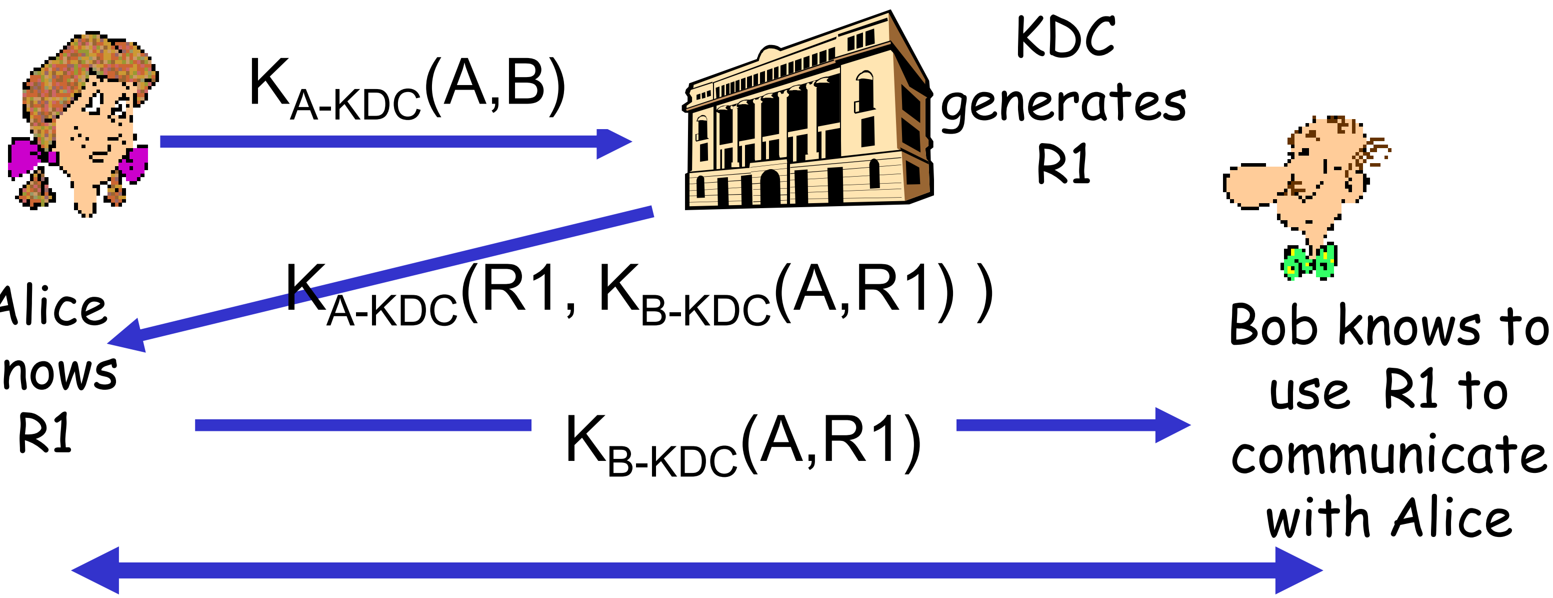
Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?

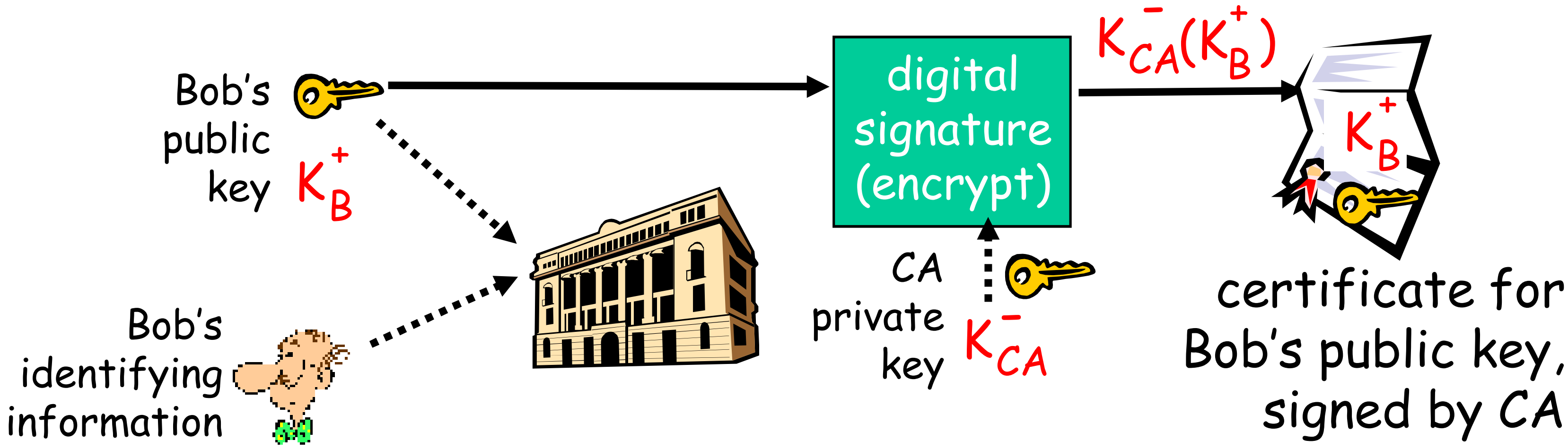


Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

Why not $R1=K_{B-KDC}$?

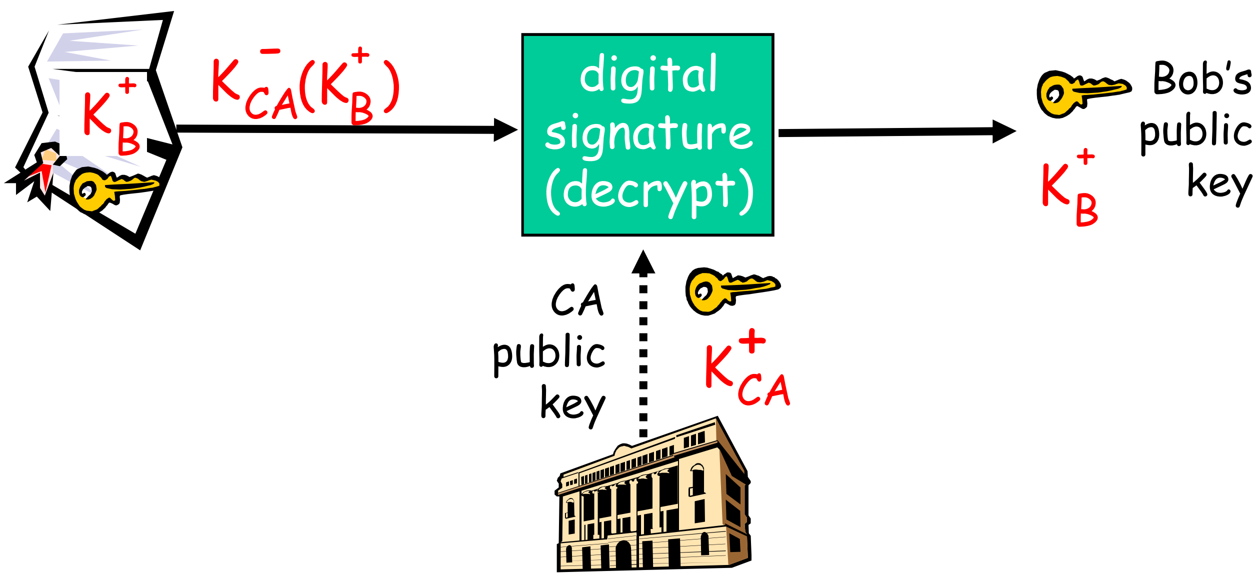
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



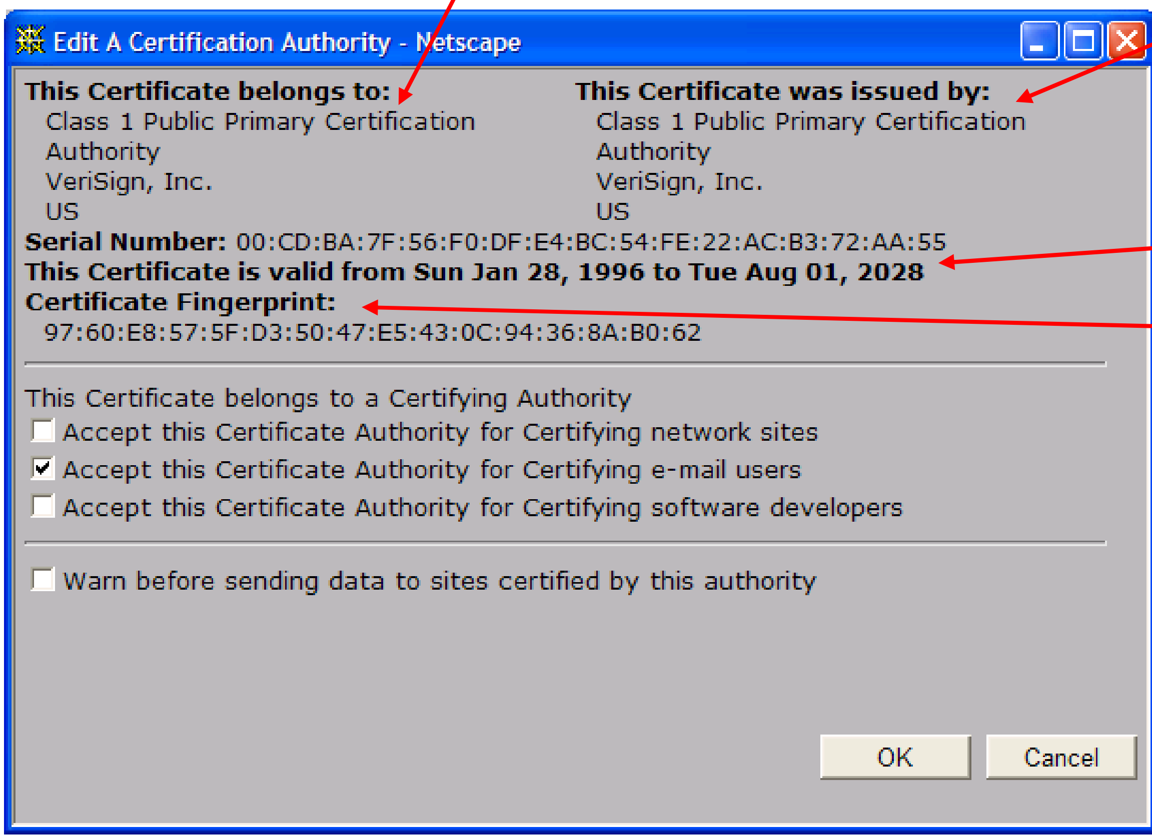
Certification Authorities

- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



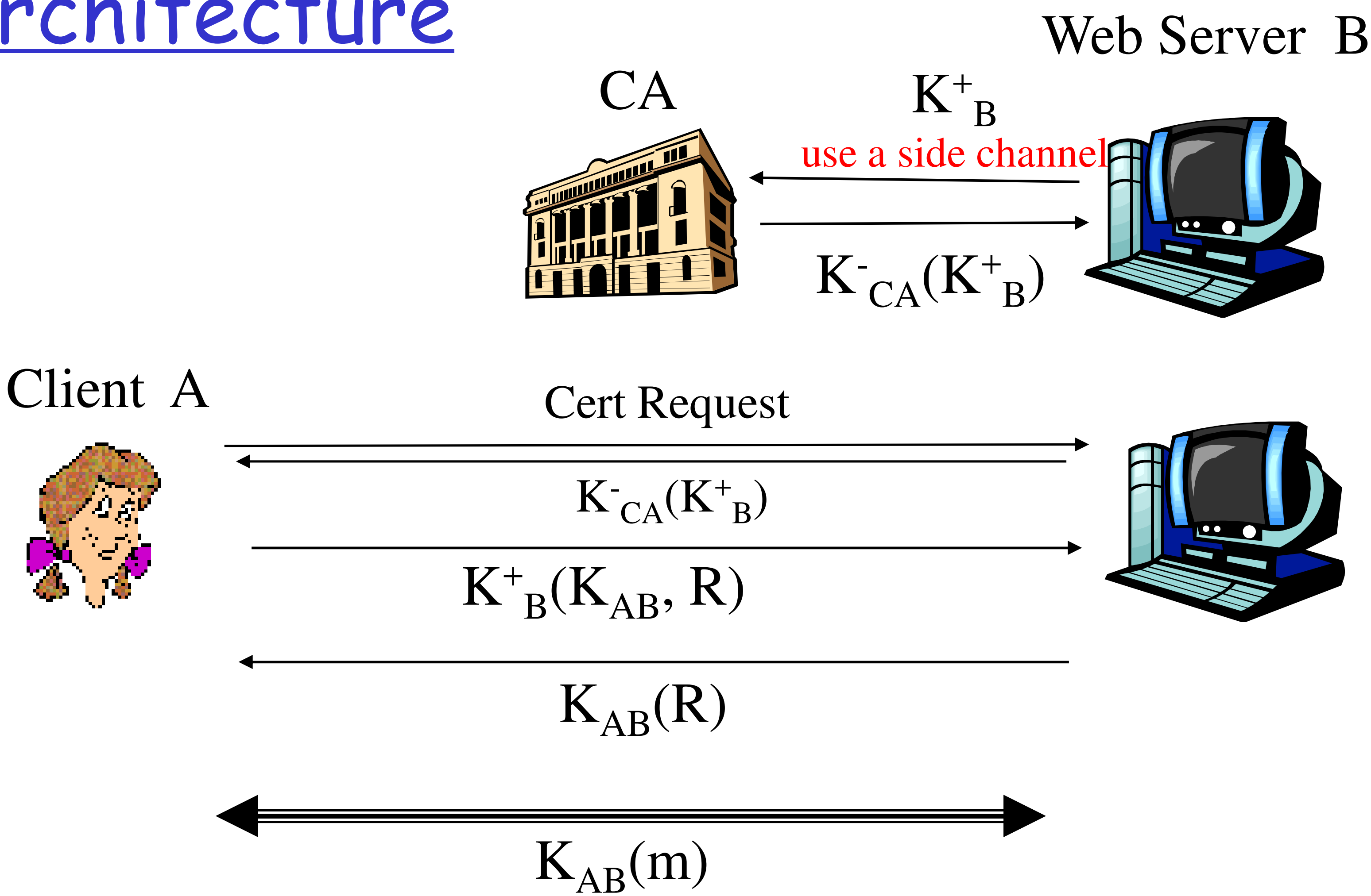
A certificate contains:

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- info about certificate issuer
- valid dates
- digital signature by issuer

Internet Web Security Architecture

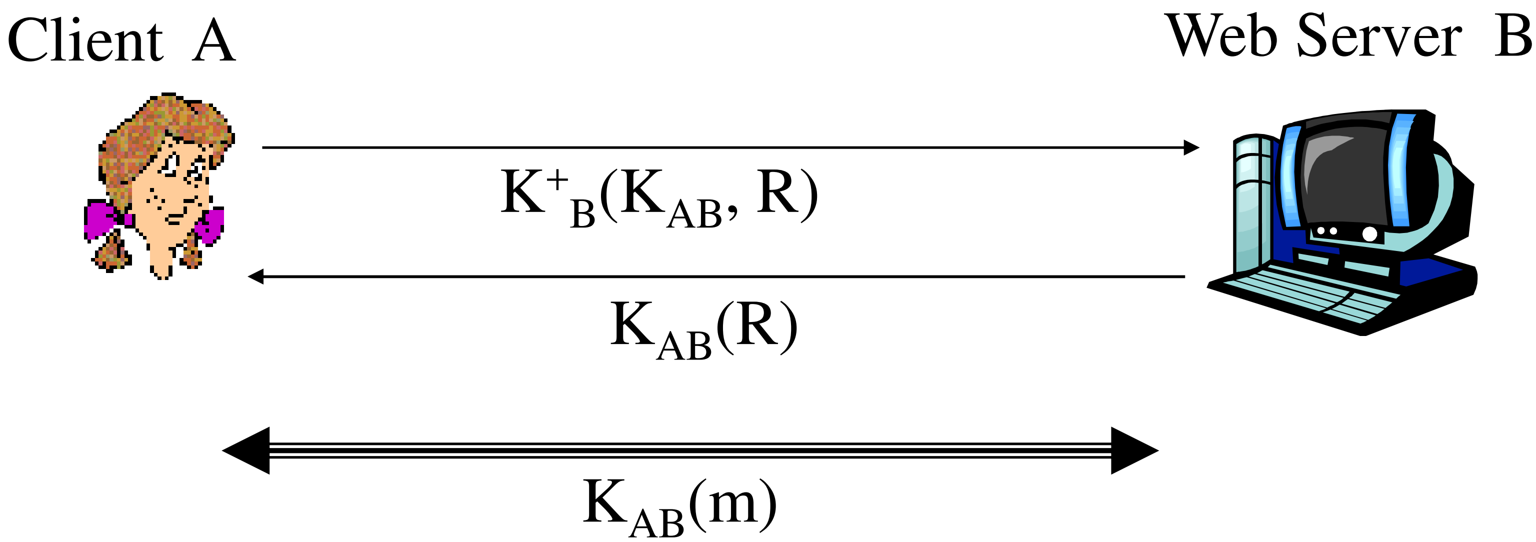


Internet Web Security Conditions

- Clients' web browsers have built-in CAs.
- CAs are trustable
- Web servers have certificates in CAs.

- Q: What if a server has no certificate?
  - Example: SSH servers

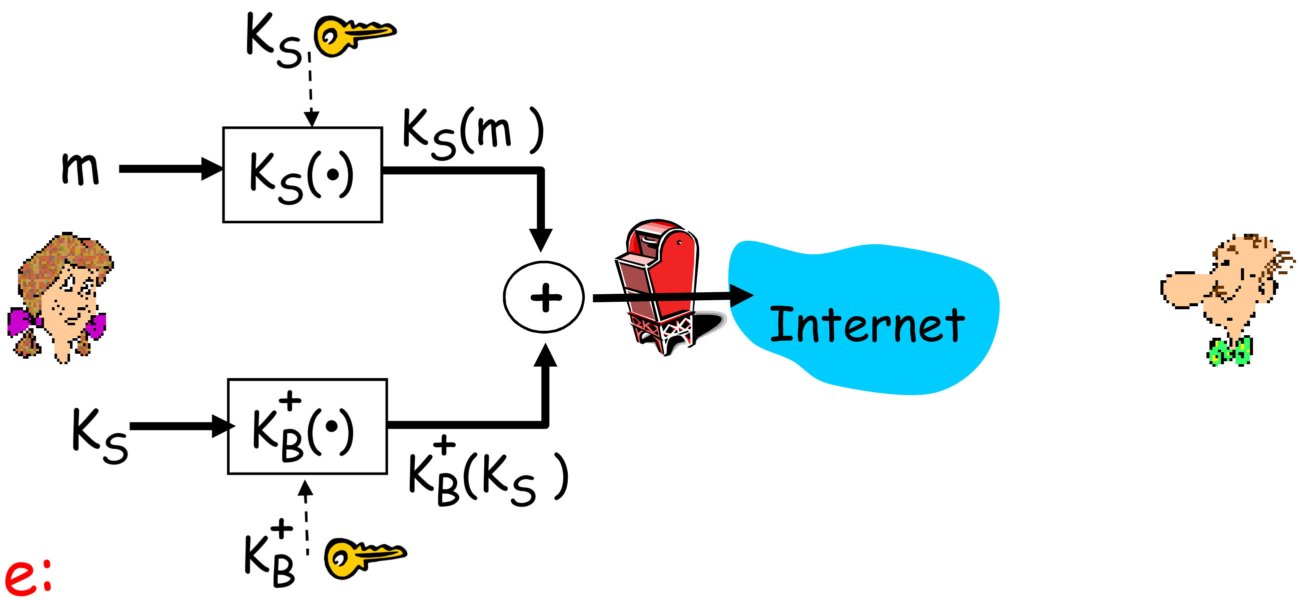
SSH Example



- Initial setup:
  - Trust the first-time connection
  - Save the server's public key

Secure e-mail

- **Assumption: Public keys are pre-distributed securely**
  - E.g: through CA, or pre-established like SSH
- Alice wants to send confidential e-mail, m, to Bob.

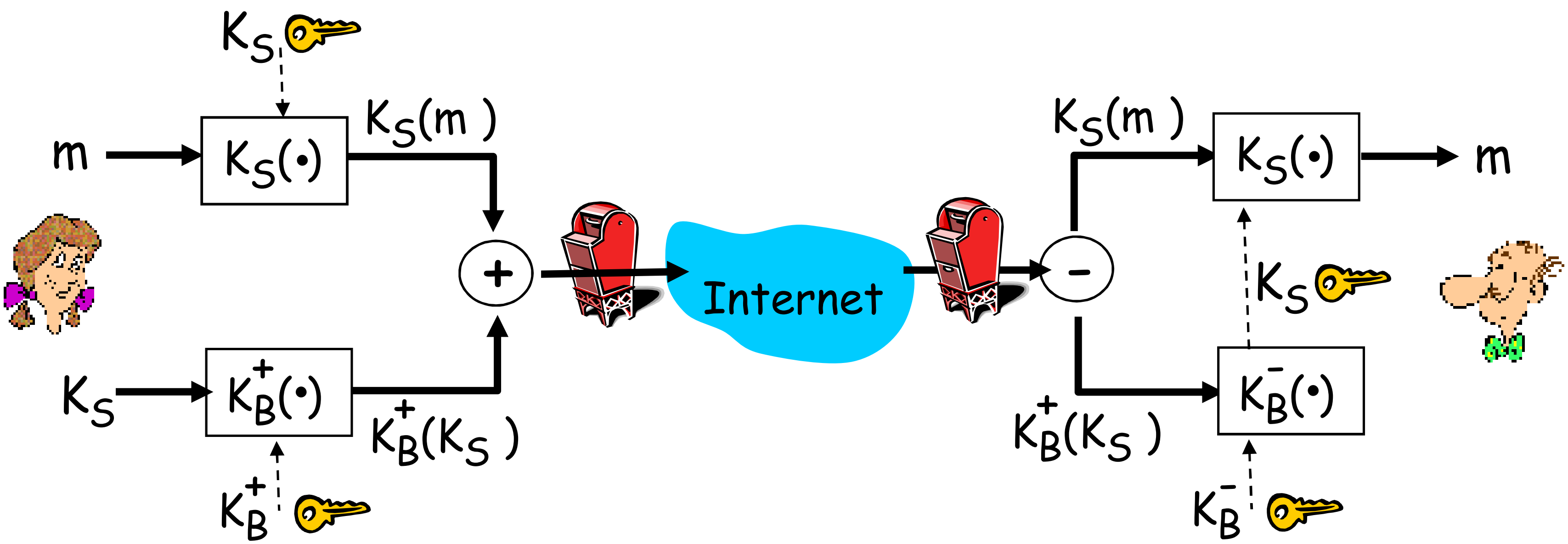


Alice:

- generates random *symmetric* private key, K_S.
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key.
- sends both $K_S(m)$ and $K^+_B(K_S)$ to Bob.

Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.

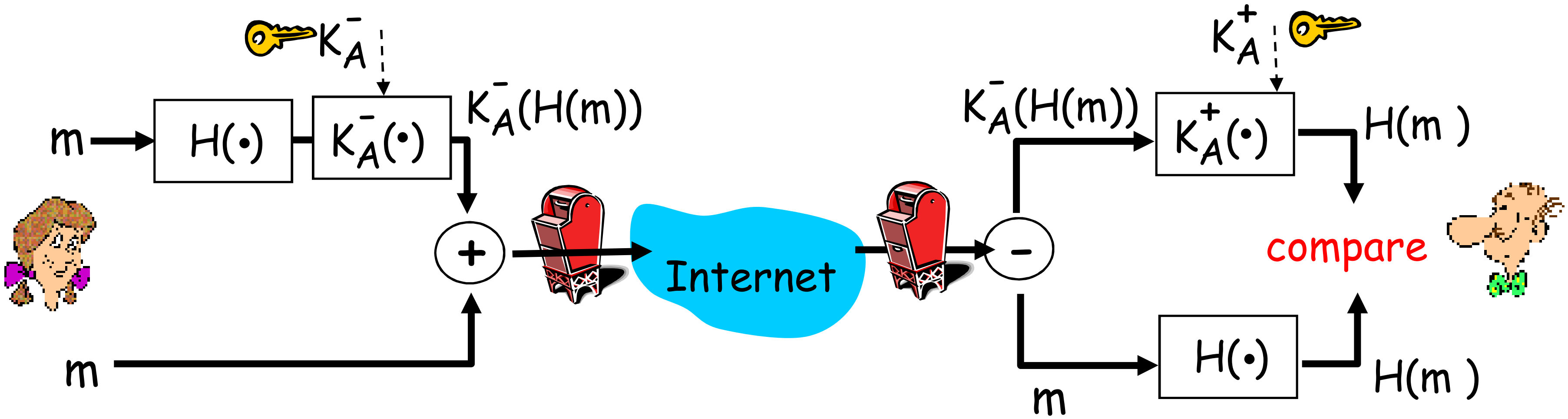


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

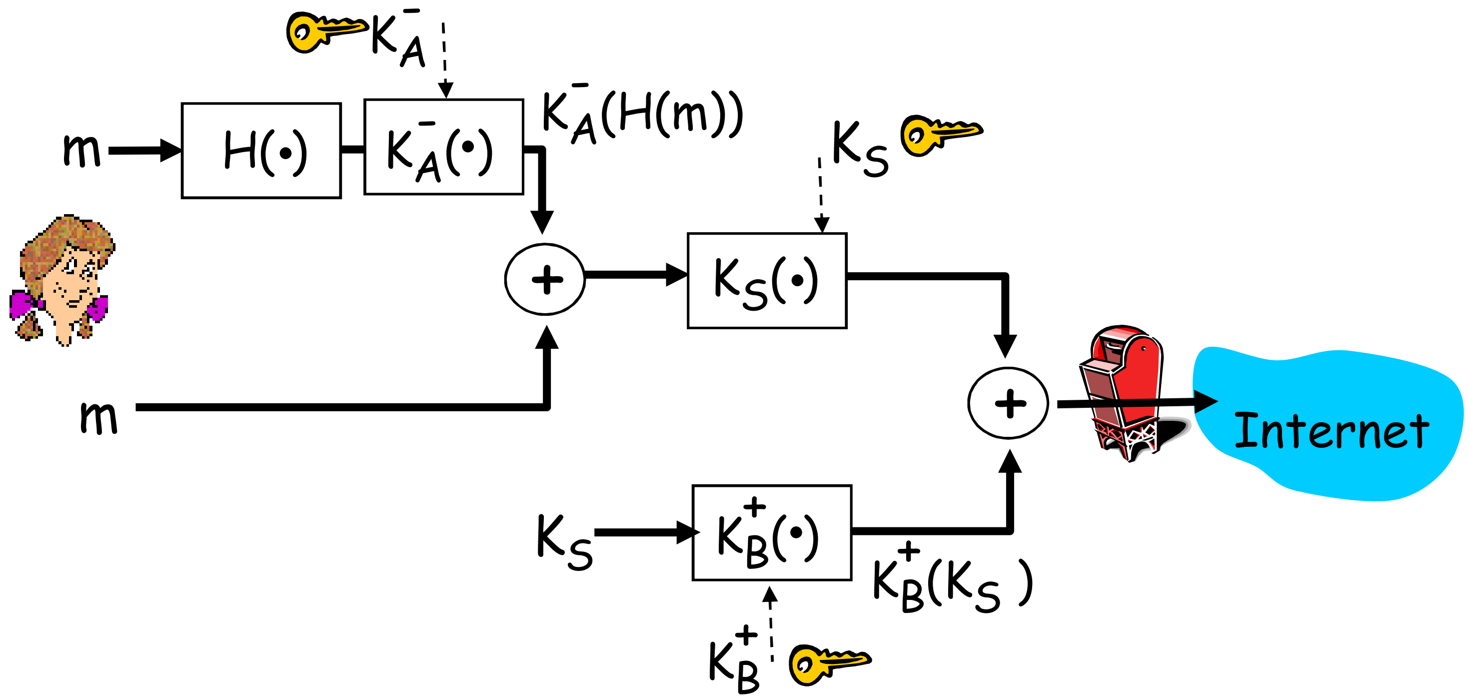
- Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key